WEEK - 14

Aim! To implement a pre-trained CNN model as a feature Extractor using transfer learnin-g models

objective:

→ to understand transfer learning and feature extraction

→ to refuse features learning from Imagent for a new task

→ to reduse training time and improve accuracy

Algorithm:

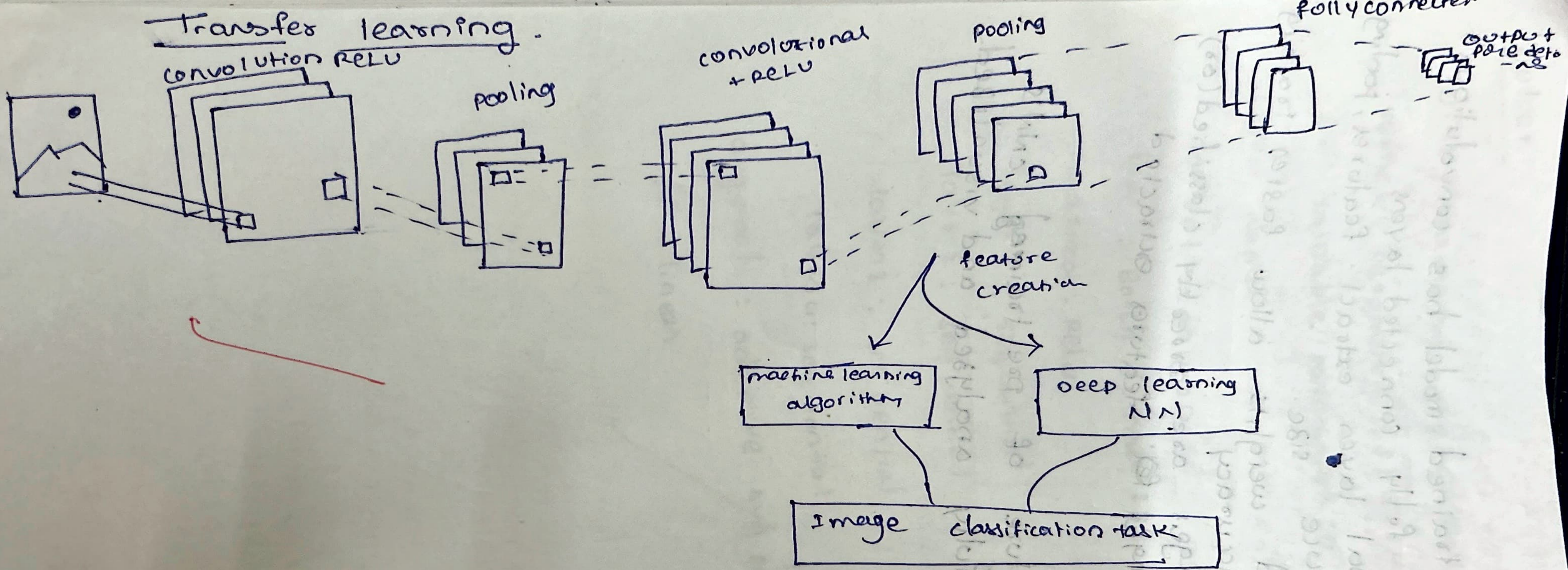
1, load the dataset (eg: cats vs doge)
2, load pre trained model
3, freeze convolutional base layer
4, Add new custom dense layer for classification
5, compile and train on small dataset
6, evalute accuracy.

pseudo code:

```
load vgg16(weights='imagnet', include-top=false)
Freeze all layers.
Add flatten () -> dense (128, relu) ->
dense (num-classes, softmax)
model = pretrained-base + New-head
compile (optimizer ='adam', loss='categorical-crosstropy')

Train on new dataset
Evalute performance.
```

Transfer learning.



observation:

→ the pre trained ResNet 18 achieved good accuracy even with few epochs.

→ training loss decreased steadily, confirming effective feature use.

→ test accuracy remained stable, showing generalization. from pre-learned features

→ model preformed well even with small dataset

Result: the pre trainied CNN successfully extracted useful features and provided high classification accuracy on a new dataset

Output:

Epoch [1/5], loss: 0.8481

Epoch [2/5], loss: 0.6266

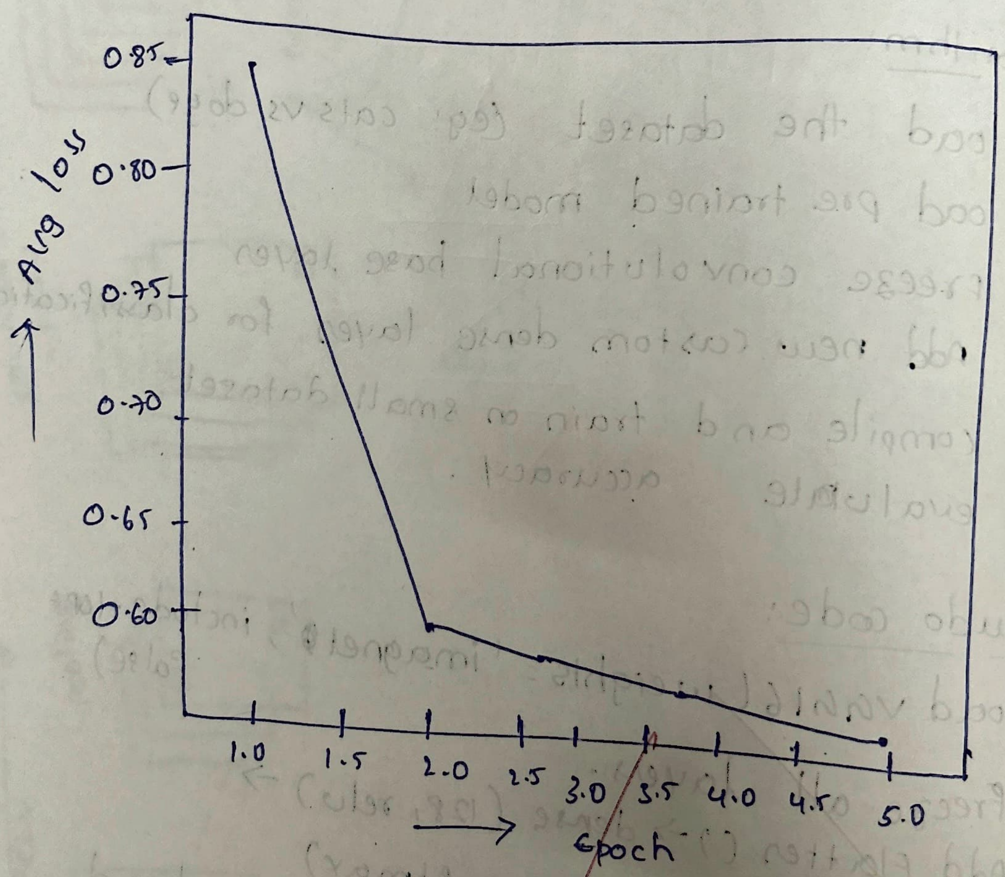
Epoch [3/5], loss: 0.5937

Epoch [4/5], loss: 0.5797

Epoch [5/5], loss: 0.5699

Training complete using pre-trained resnet feature extraction

Training loss curve - Transfer with Re



```python
import tensorflow as tf
from tensorflow.keras import layers, models
import matplotlib.pyplot as plt
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.cifar10.load_data()

x_train, y_train = x_train[:10000], y_train[:10000]
x_test, y_test = x_test[:2000], y_test[:2000]
x_train = x_train.astype("float32") / 255.0
x_test = x_test.astype("float32") / 255.0

base_model = tf.keras.applications.MobileNetV2(
    input_shape=(64, 64, 3),
    include_top=False,
    weights='imagenet'
)
base_model.trainable = False  # freeze layers
model = tf.keras.Sequential([
    layers.Resizing(64, 64),
    base_model,
    layers.GlobalAveragePooling2D(),
    layers.Dense(64, activation='relu'),
    layers.Dense(10, activation='softmax')
])

model.compile(
    optimizer='adam',
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)

history = model.fit(
    x_train, y_train,
    validation_data=(x_test, y_test),
    epochs=2,
    batch_size=64,
    verbose=1
)
loss, acc = model.evaluate(x_test, y_test, verbose=0)
print(f"\n✅ Test Accuracy: {acc:.4f}")
plt.figure(figsize=(10,4))
plt.subplot(1,2,1)
plt.plot(history.history['accuracy'], label='Train Acc')
plt.plot(history.history['val_accuracy'], label='Val Acc')
plt.title('Accuracy')
plt.xlabel('Epoch')
plt.legend()
plt.subplot(1,2,2)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Val Loss')
plt.title('Loss')
plt.xlabel('Epoch')
plt.legend()
plt.tight_layout()
plt.show()
```

```
/tmp/ipython-input-1693943737.py:18: UserWarning: `input_shape` is undefined or non-square, or `rows` is not in [96, 128, 160, 192, 224]. Weights for input shape (224, 224) will be loaded as the default.
  base_model = tf.keras.applications.MobileNetV2(
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/mobilenet_v2/mobilenet_v2_weights_tf_dim_ordering_tf_kernels_1.0_224_no_top.h5
9406464/9406464 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
Epoch 1/2
157/157 ━━━━━━━━━━━━━━━━━━━━ 43s 239ms/step - accuracy: 0.3827 - loss: 1.8126 - val_accuracy: 0.6035 - val_loss: 1.1824
Epoch 2/2
157/157 ━━━━━━━━━━━━━━━━━━━━ 40s 257ms/step - accuracy: 0.6609 - loss: 1.0089 - val_accuracy: 0.6180 - val_loss: 1.1267

✅ Test Accuracy: 0.6180
```